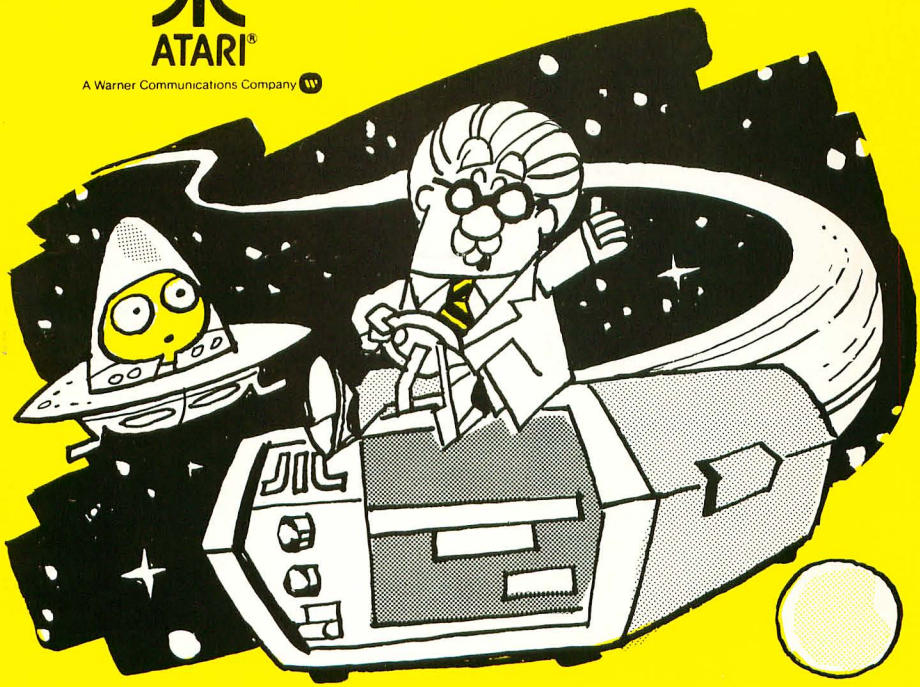
Educational Software inc. PRESENTS

TRICKY TUTORIAL(TM) #7

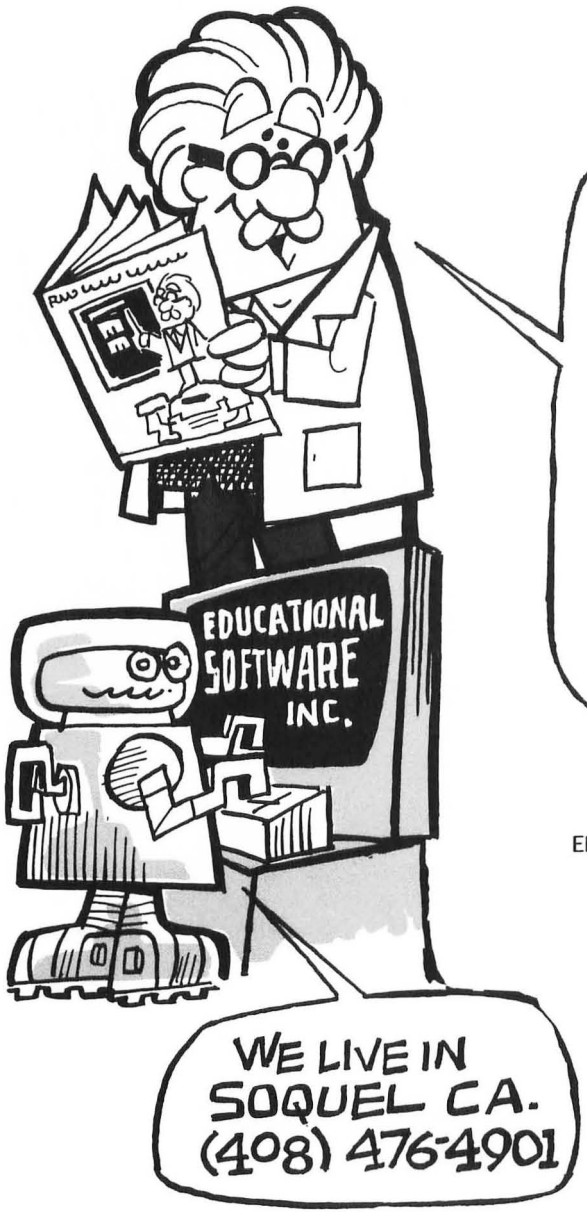
DISK UTILITIES

FOR ATARI® DISK DRIVES

A Warner Communications Company



DISK DRIVES MADE EASY !



We have many other fine Tutorials teaching you how to use your ATARI!

My pal PROTOTYPE and I have also written a nice book, called the MASTER MEMORY MAP that will unlock most of the machine's mysteries for you!

See your local DEALER or call for a catalog

EDUCATIONAL SOFTWARE inc.

4565 Cherryvale Ave.

Soquel, Ca. 95073

EDUCATIONAL SOFTWARE inc.

presents

TRICKY TUTORIAL #7

DISK UTILITIES

by

Jerry White

How to Load

These programs require 32K RAM memory, an ATARI 810 Disk Drive, and the ATARI BASIC Language Cartridge. First, insert the ATARI BASIC Language Cartridge in the (Left Cartridge) slot of your computer. To load and run the disk, turn on your disk drive. When the busy light goes off, insert the disk. Power up your computer and turn on your video screen. When the MENU appears, select the utility you want. Later, after you are familiar with the Utilities, take the ones you need and copy them to your own disks.

M E N U

HOW TO USE MENU:

The MENU program was designed to provide a screen display of disk files and run BASIC programs. MENU will work with either ATARI DOS 1 or DOS 2.0S and RUN BASIC programs that are in SAVED or tokenized format.

MENU will automatically LOAD and RUN when your Tricky Tutorial diskette has been used to "BOOT UP" your ATARI computer. Any time you want to have MENU run first on your disks, just use the COPY function from the DOS menu to put Menu and AUTORUN.SYS from this disk onto yours.

Booting up means that when a disk has MENU on it, and an AUTORUN.SYS that calls Menu, Menu is run by just turning on your ATARI 810 disk drive (and other peripheral equipment, if any: printer, interface, TV, etc). Insert the diskette into Drive #1 and shut the door. Make sure your ATARI BASIC cartridge is installed, then just turn on the computer.

ATARI's DOS 2.0S will load it's File Management Software, then look for an AUTORUN.SYS file. Since your diskette does contain an AUTORUN.SYS file, DOS will LOAD and execute the instructions in this short program. This particular AUTORUN.SYS tells the computer to say "HELLO", then RUN a program called INTRO. INTRO displays our cover message and runs the MENU program. A sequence of programs that RUN other programs is called CHAINING. More on that later.

MENU begins by reading what is called the Disk Directory File. This file contains a list of every file on a diskette. As MENU reads this information, it will count the number of files read, and display the current count on the screen. Once all files have been read, the screen will also display the number of FREE or available sectors left on the disk.

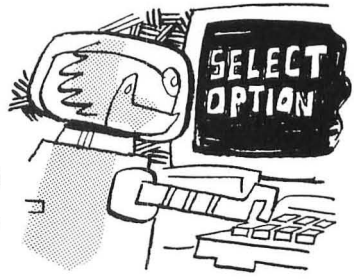
At this point there will be a brief delay as MENU does some internal calculating. Once this step has been completed, The authors name will be replaced by the words, "PRESS START". When the START key is pressed, the screen will appear as follows:

SELECT OPTION

PRESS START

RUN BASIC PROGRAM

DISPLAY DIRECTORY



Note that the words, "PRESS START" and "RUN BASIC PROGRAM" are in WHITE. Throughout this program, the currently selected option will always appear in WHITE. This will indicate what the computer will do if you PRESS START.

The OPTION and SELECT keys are interchangable. They are used to SELECT the OPTION you desire. Press one of these keys now then release it. Note that there were two sounds. One was heard as you pressed the key and the other as you released it. The program acted upon your request as you released the key. Note that the words, "RUN BASIC PROGRAM" are no longer white but "DISPLAY DIRECTORY" has changed to white. Now press OPTION or SELECT again. Once you released that key, the screen returned to it's original colors.

Now that you understand what Menu is, let's see how easy it makes choosing programs on your disks. Press SELECT (or OPTION) so that DISPLAY DIRECTORY is white, then press the START key.

You have told MENU to display all files on the diskette. The screen is set up to display 9 files. If more than 9 files are on the disk, they will be displayed, up to 9 at a time, each time you PRESS START. Once all files have been displayed, you will be returned to your original two choices. Again, RUN BASIC PROGRAM will appear in white.

PRESS START and a new screen will appear. Note that there are now options numbered 1 thru 9. Only numbers 4 thru 9 are BASIC programs. Number 1 is CONTINUE. This would be selected to do just that, continue. In other words, you would be saying, "I don't want to return to BASIC (#2), or call DOS (#3), or run any of the programs numbered 4 thru 9.

If you just wanted to use BASIC for working on a program, you would select number 2. If you wanted to use DOS, you would select number 3. If you wanted to RUN one of the programs numbered 4 thru 9, you would select it's number.

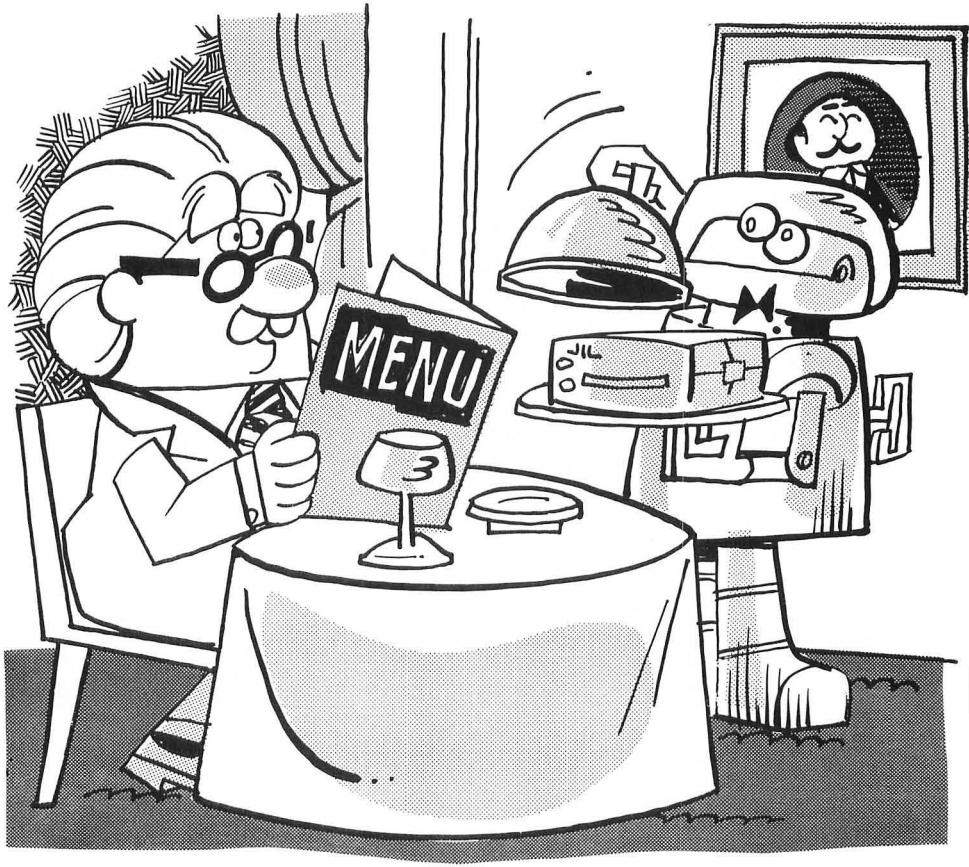
You may select a number by pressing the OPTION or SELECT key until the desired option appears in white. Note that here, you do not have to release the button for the option to change. Try it. Just hold down the OPTION key. One by one, each option will turn white. When you're done fooling around, leave the screen so that CONTINUE is white. Now press the START key.

If there are additional programs to display, a new batch of options will appear. In each case, the first three will remain the same, and up to 6 new options (BASIC programs) will appear. When the last program has been displayed, and the CONTINUE option is selected, MENU will redisplay the first 6 BASIC programs.

I said there were two methods of selection. The other is pressing the number key of the desired option. There is no need to press RETURN or START when a number key is used. As soon as a valid key is pressed, MENU will respond accordingly.

Remember that when you displayed the DISK DIRECTORY, you saw files such as DUP.SYS and AUTORUN.SYS? But when you used the RUN BASIC PROGRAM option, these files were not displayed. That is because this MENU program assumes that if a filename has any extention, (such as .SYS), it is not a BASIC program. Conversely, when no extention is found, MENU assumes the file to be a BASIC program in SAVEd format.

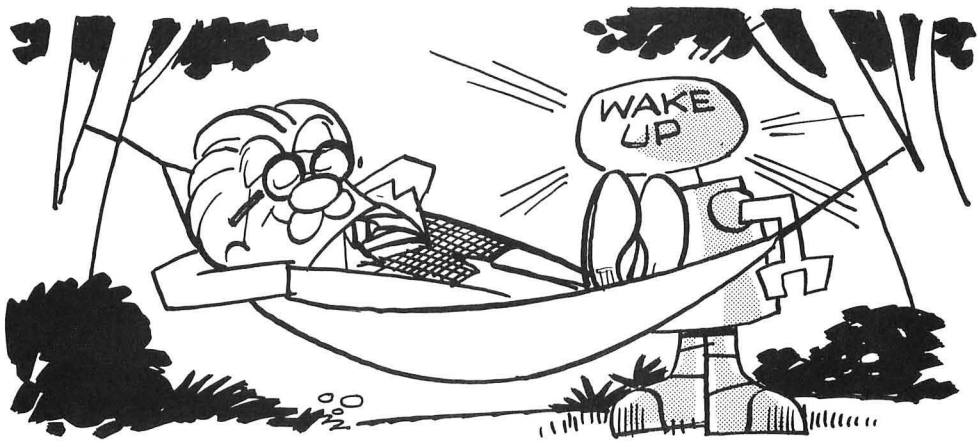
You may use MENU on all your standard ATARI diskettes for convenient directory displays and BASIC program selection. You will just have to remember to use filename extentions in all NONBASIC program names. You will also find that if you use an extention, that filename will NOT appear in your BASIC program selection, even though it might be a BASIC program. In other words, DO NOT use extentions such as .BAS to indicate a BASIC program. We will discuss standards for filenames and extensions a bit later on.



FORMAT1 - A Disk Formating Aid

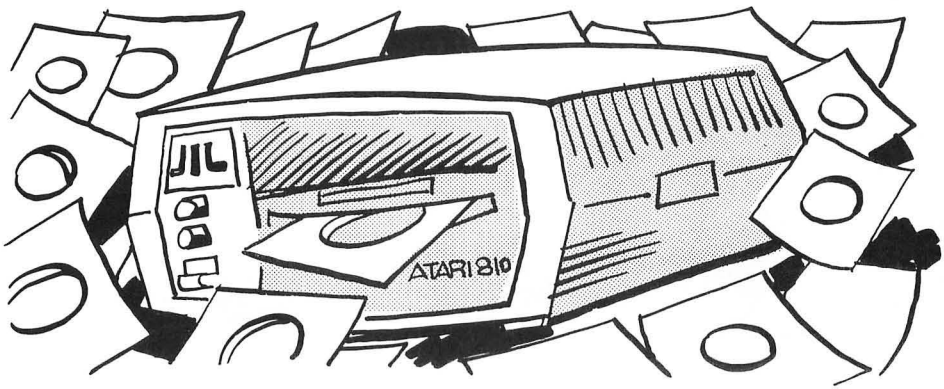
HOW TO USE FORMAT1:

Why have a BASIC program to format disks when DOS does the same thing?Well, a Professor's life is very busy with all these TRICKY TUTORIALS to write. When I get a chance to nap, I DO IT!



So anyway, while I was laying around, er, I mean working, one day, I realized that when DOS is used to format diskettes, the user must type 6 keys before the formatting begins. With FORMAT1, all you have to do is insert the disk and press the START button.

BIG DEAL! To format one disk, using FORMAT1 saves 5 keystrokes. If you just bought a box of 10, you save 50 keystrokes. If you format disks in even larger quantities, and have two drives, you could remove the PRESS START prompt, make minor modifications to allow you to flip floppy between the two drives, with no keys to press at all!



HOW TO USE DISKLIST

The DISKLIST program reads the disk directory file from any ATARI diskette and generates a disk jacket label on any compatable 40 or 80 column printer. This label will contain a heading, the names of each file on the diskette, and the number of free blocks or sectors.

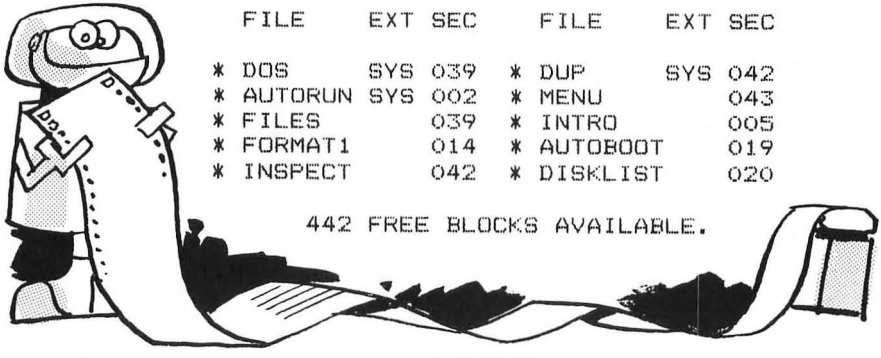
You may use plain white paper, then cut out the listing and attach it to the protective outer jacket using rubber cement or scotch tape. This will help you to identify your diskettes at a glance. It is recommended that these labels be used only for diskettes which are nearly full. Naturally, if the contents of your diskette changes, the label would have to be reprinted.

When you RUN this program, you will be prompted to insert the appropriate diskette, then enter the drive number you are using.

You may enter a heading for your label. You may enter up to 40 characters of descriptive information, or just press RETURN to bypass this option.

Once this has been done, the program will read your diskette and print your label. You will then have the option to return to the MENU program by pressing the OPTION button, or rerunning DISKLIST by pressing the START button.

Here's a example of output from DISKLIST:



HOW IT WAS WRITTEN:

The program itself is short and stright forward. It simply opens the disk directory to read all files using wild cards. For example, if you specified disk drive number 2, the directory file would be opened as follows:

```
OPEN #1,6,0,"D2:*.*"
```

The data between quotes is stored in the string F$. As each record is read, it is checked to see if it's length is less than 5, or if positions 5 thru 8 contain the letters, "FREE". If the diskette is in DOS 2 format, the last record will contain the word "FREE". If the diskette is in DOS 1 format, the last record will contain only the number of free sectors. The length of that number will be less than 5, and DISKLIST will know we have reached the end of the directory file.

IOCB (Input Output Control Block) #2 is used for your printer. Directory data is read into the string A$, and printed using a ? or PRINT #2 command.

The resulting printed material will usually fit quite nicely on the disk jacket unless you have a large number of files on your diskette.

HOW TO CREATE AN AUTORUN.SYS FILE

When an ATARI computer is turned on, the Operating System (O.S.) performs many initialization tasks. One of these tasks is to find out if a disk drive is available. If so, it looks for instructions by reading the first sector on whatever diskette it finds on drive one.

Using your Tutorial diskette as an example, the first 3 sectors (boot sectors) are read. Then the file called DOS.SYS is read and the O.S. sets up ATARI's F.M.S. or File Management System. Once this has been done, DOS looks to see if a file named. AUTORUN.SYS is on the diskette. If not, control of the computer is turned over to a cartridge, in this case, ATARI BASIC.

```
AUTORUN.SYS CREATE PROGRAM FOR BASIC
          DOS VERSION 2

  UPDATED 3/25/82  by Jerry White

BOOT INTERFACE (Y or N)?N

ENTER COMMANDS:

?RUN"D:MENU"

OPENING   D1:AUTORUN.SYS

CREATING D1:AUTORUN.SYS

CLOSING   D1:AUTORUN.SYS


READY
```

If the diskette contains an AUTORUN.SYS file, the system will execute the instructions it receives as a result of reading this file (these instructions may be as simple as RUN"D:MYPROG", or more detailed if you need). Assembler instructions may be executed immediately as each instruction is read. BASIC instructions will be executed after the entire AUTORUN.SYS file has been read. See your DOS 2 Manual or De Re ATARI if you desire further technical information.

The AUTORUN.SYS file is a handy little tool which may be used to automatically perform user specified instructions such as RUN a program. The AUTORUN.SYS file on this EDUCATIONAL SOFTWARE diskette tells your computer to set the foreground color so that it equals the background color immediately. Why? This was done to prevent the READY prompt from appearing in the upper right corner of the screen. Why? It is my humble opinion that the READY prompt should tell the user that the system is READY to use. In this case, it would not be true.

On this disk, the AUTORUN instruction is RUN"D:INTRO". Our obedient computer will look for and hopefully find a program named INTRO, then LOAD and RUN it. The INTRO program simply displays introductory information, tells you that the MENU program is being LOADed, and RUNs it for us. This could be describd as CHAINING programs. DOS calls AUTORUN.SYS, AUTORUN.SYS calls INTRO, and INTRO calls MENU. For your programs, you may want to create an AUTORUN.SYS on disks holding your favorite games or business programs. AUTORUN.SYS will automatically run them. What to do if you have several games on a disk?--use MENU and AUTORUN.SYS.

The AUTOBOOT program may be used to create your own AUTORUN.SYS file on any DOS version 2 diskette. DO NOT RUN AUTOBOOT until you have made a backup copy of your EDUCATIONAL SOFTWARE diskette. The AUTORUN.SYS file on this disk is locked to prevent accidentally writing over it.

As an example of using the AUTOBOOT program, let us assume you have decided to use my handy little MENU program and would like to have MENU RUN automatically, bypassing the INTRO program. O.K. You would begin by using DOS to FORMAT a blank diskette. Once formatted, use DOS option H to write DOS files. Be sure to use DOS 2.0S. When in doubt, simply BOOT or reboot your computer using our diskette (which has DOS 2.0S on it), then load DOS from the MENU program, (see MENU instructions). Now use DOS option O to DUPLICATE the MENU program.

If all went well, you should now have three files on the new diskette. Check this by using DOS option A to display the Disk Directory. Type a capital A, then press RETURN twice. The first time you press RETURN is to enter the A option. The second time is to bypass all options of the Disk Directory feature. This tells DOS to simply display all files found in the Disk Directory file onto the screen. There should be three; DOS.SYS, DUP.SYS, and MENU. The number to the right of the filenames are the number of disk sectors occupied by that file. The last line tells you the number of FREE or unused sectors.

Now to create our AUTORUN.SYS file. Before we can RUN the AUTOBOOT program, we must exit from DOS. To do this, type B and press RETURN. Since the BASIC Cartridge is in place, you will now see the familiar READY prompt. Put your TRICKY TUTORIAL diskette back into drive one, type RUN"D:AUTOBOOT", and press RETURN.

Before you go any further, remove the TUTORIAL diskette and insert the one on which we will write our new AUTORUN.SYS file. The first prompt will ask you if you wish to BOOT INTERFACE (Y or N). If you will be needing the serial ports of ATARI's 850 Interface Module, respond Y. If you don't need them, or if you don't have one, respond N. There is no need to press RETURN.

At this point we can enter whatever BASIC commands we want to have executed by AUTORUN.SYS. You can only type in one line of BASIC commands, so use colons and keep the total length to 3 lines on the screen (133 characters, as in your BASIC Manual, see last page). The AUTOBOOT program will automatically generate code to set the background and foreground colors equal (eliminating the READY prompt). By the way, the prompt will be on the screen, you just won't see it. I mention this to point out that if you were going to use a PRINT command, you will want the text to show up on the screen. For this reason you should begin your AUTORUN instructions with a GRAPHICS command to set the colors back to normal. To keep this example simple, just type RUN"D:MENU" and press RETURN.

As the AUTOBOOT program RUNs, it will tell you what it is doing. It will OPEN, WRITE, then CLOSE a file named AUTORUN.SYS. Upon completion, AUTOBOOT will simply END. All you need to do to see your AUTORUN.SYS file do it's thing is shut off the computer, then turn it back on again. The computer will do the BASIC statement you gave it.

Some people say that frequently turning electronic devices off and on is not good to do, since it wears out components. To avoid this and still test your AUTORUN.SYS file, just type POKE 580,1 and press RETURN, then press SYSTEM RESET. This will cause a cold start, similar to turning off and then on again.

You may enter a series of unnumbered BASIC commands into the AUTOBOOT program. They must be separated by a colon and not exceed three lines on the screen. You might wish to display a message, reset margins, or simply create a disk that will automatically call DOS.

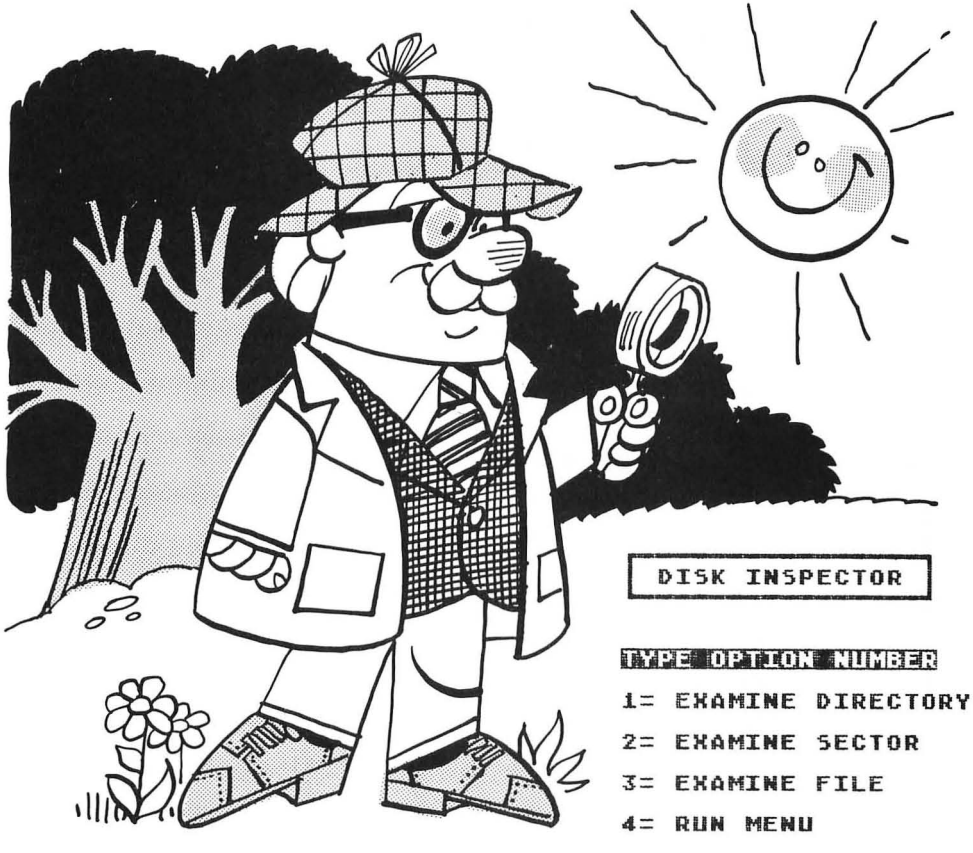
If no BASIC Cartridge is found, DOS will load and display it's own MENU of options, ignoring the AUTORUN.SYS file. To automatically use DOS options with the BASIC Cartridge installed, use AUTOBOOT and simply enter the command DOS.

The AUTOBOOT program provides a method of automating tasks that might otherwise have to be done manually. Isn't automation wonderful?

INSPECT

HOW TO USE INSPECT

RUN the INSPECT program. The options display identifies the program by it's full name, "DISK INSPECTOR", and provides four options. When entering an option number, there is no need to press the RETURN key.



1 - EXAMINE DIRECTORY

Let's begin by typing the number 1 to examine the directory file of this diskette. The screen should now display:

S	FN	FILE	EXT	SSEC	NSEC
L	0	DOS	SYS	4	39
L	1	DUP	SYS	43	42
L	2	AUTORUN	SYS	85	2
L	3	MENU		87	43
L	4	FILES		130	39
L	5	INTRO		169	5
L	6	FORMAT1		174	14
L	7	AUTOBOOT		188	21
L	8	INSPECT		209	42
L	9	DISKLIST		251	20
L	10	RPMTEST		271	14

The first column provides the Status of each file. In this case, each file shows the status L. The L indicates that the file is locked. If the file were unlocked and available, the status would be blank. If a file was Deleted, the status would be indicated as the letter D.

The second column shows the File Number. The first file number is zero. The maximum number of files that may be stored on a given diskette is 64. If the directory were completely full, the last file number would be 63.

Following the file number is the filename and extention if any. Note that the "." between filename and extention is NOT displayed.

Finally we have two columns of decimal numbers. The first is "SSEC" (the starting sector) and the second is "NSEC" (number of sectors). Note that our first file, DOS.SYS, begins at sector 4 and occupies 39 sectors. The first 3 sectors on the disk are reserved as BOOT sectors.

At the bottom of the screen is the message, "PRESS ANY KEY FOR OPTIONS". Do this and you will return to our original four options.

2 - EXAMINE SECTOR

This time type the number 2 to EXAMINE SECTOR. The program will then ask us to "TYPE SECTOR NUMBER RETURN?". Let's examine the first sector of the AUTORUN.SYS file. Type the number 85 then press RETURN. The program will then ask us to TYPE 1 FOR CHARACTER OR 2 FOR HEXDUMP. Type the number 2 (NO DEPOSIT, NO RETURN).

Each sector consists of 128 bytes. Note that the first two bytes of the AUTORUN.SYS file are FF (decimal 255). This indicates that the file is in machine language or what is also known as Object Code. Once again, press any key, to return to our options display.

Now let's examine sector 87 which is the beginning of our MENU program. Note that the first 3 bytes are 00. This indicates a BASIC program in SAVEd format.

Return to the options display and have another look at sector 87, but this time specify 1 for character format. Note that the first three bytes are hearts. The heart character is ATASCII character 0.

3 - EXAMINE FILE

Get back to the options display and type the number 3 to examine a file. The program will then ask us to ENTER FILENAME? Type MENU then RETURN. Once again we see the beginning of the MENU program but in a somewhat different screen layout. Here we begin with byte 1 instead of 0, just to say that this is the first byte of the MENU program. For each byte, the Decimal and Hex Values are displayed along with the ATASCII character representation.

At the bottom of the screen you will see, "OPTION=OPTIONS START=CONTINUE". To continue looking into the MENU program, (20 bytes at a time), press the START button. When you are ready to return to our options display, press the OPTION button.

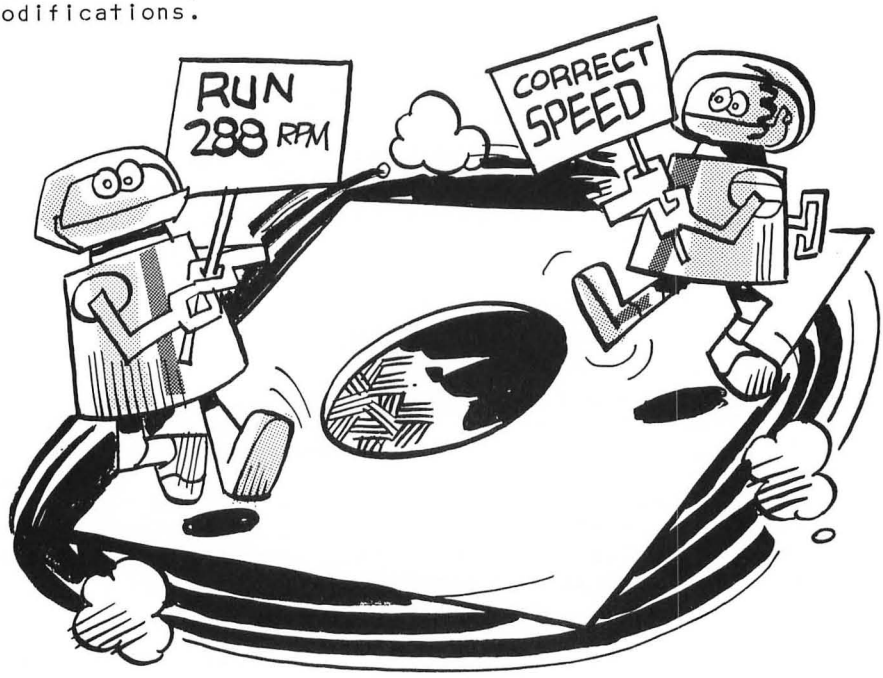
4 - RETURN TO MENU

The fourth and final option is used to return to our main MENU program.

RPMTEST

If you suspect that you are experiencing problems caused by your 810, this little program might come in handy. It will allow you to check the speed of your disk drives.

Before we go any further, I'd like to thank the author of this program, Bob Christiansen of Quality Software, for his permission to include his program in this package. RPMTEST was originally published in the May 1982 issue of COMPUTE! magazine. Although my version is slightly different, the logic, and the machine language routine remain the same. We cannot reprint the COMPUTE documentation, nor do we wish to steal it by making minor modifications.



RUN this program now. If your drive is running at the proper speed, your diskette should spin between 285 and 290 times per minute. The absolute perfect speed is said to be 288.

The question that the program asks you is: "Which drive number do you want to test"? Before you answer, make sure a formatted diskette is ready in the appropriate drive. Note that this program will test by reading sector number one of the diskette, 100 times. You can use any program disk without fear of losing data or damage to the disk. Enter the appropriate drive number from 1 thru 4, then press RETURN.

It will take about 22 seconds for the test. The speed will then be displayed. If the speed is within normal limits, the background color of the screen will change to green, and a message will say, "DRIVE SPEED IS O.K.". If your drive speed is too fast or too slow, a message will be displayed to that effect, and the background color of the screen will be red.

At this point, you may test again by pressing the START key, or return to the MENU program by pressing the OPTION key. If you decide to return to the MENU program, make sure our tutorial program diskette is ready in drive one before you press the OPTION key.

When testing the speed of any drive, it is important that the diskette you use is spinning freely. If you hear any unusual sounds, the diskette might be rubbing against it's jacket, causing the RPM speed to be slow. you may want to test the drive speed using several different disks.

It is also important to note that the speed of the 810 has been known to be about 3 RPM faster when it is cool, than after it warms up. Since most of the time the drive will be warm, it should be tested under those conditions.

If after running this test a few times, you find your drive to be consistantly fast or slow, the speed should be adjusted. You can do this yourself as described in the COMPUTE article, or have an authorized repair center do it for you. Note that if you decide to do it yourself, you will be voiding your warranty. The choice is yours. I am not describing the details here to protect the hundreds of you who would do something wrong, thereby breaking their drives. The speed is adjusted inside the drive by turning a small wheel on a "Pot", but many drives have this pot sealed so that if you turn them too hard they break. My pals and I found this out late one night in our labratory, and it took days to get a replacement part!

If you find that the speed varies by more than 2 or 3 RPM from test to test, you have a problem that should definately be brought to the attention of an authorized service center.

ATARI DISKFILES TUTORIAL

Many new computer owners are anxious to learn how to write useful programs for themselves. After reading the literature packed with the machine, the new owner is often overwhelmed by the many technical aspects of using a computer. Since you can not learn any programming language overnight, a seemingly endless period of trial and error usually follows. A new owner is often seen burning the midnight oil and arguing with a defenseless TV or monitor.

If he or she perseveres long enough, useful programs can be written. The new programmer is now ready for bigger and better things.

Assuming that a disk drive is available and our "hacker" has had some experience with DOS and the loading and saving of programs, he or she is ready to write some kind of database program. Now don't be scared off by a big word like "DATABASE". It just means you are going to save some useful information on a disk for later use. Our examples will be simple, but the methods will work just as well for more complex data.

The data file may consist of a simple list of record albums for a start. When you have gained experience, you might want to try a Personal Finance System. If you are at the point in your programming career to want to keep records, or think you might be in the near future, read on.

Start with something very simple. Don't try to write that Financial Package yet. There is a lot to learn first about file structure and I/O. I/O stands for Input/Output. Input is data being read by a program. Output is data being produced by a program. A file consists of one or more records, and a record is an item within a file. Records may be broken down further into fields. We will be using simple records containing a single 10 character field as our record, and create a sample 10 record datafile.

WHAT THE HECK IS A DISK FILE?

A FILE is a collection of related data. Files fall into two major catagories; Program files, and Data files.

Each major section within a file is called a RECORD. In a data file, each record may be divided into a number of FIELDS.

For example, let's assume you had a data file of names and telephone numbers. The name could be one field called NAME$, and the telephone number could be a field called PHONE$. Using IOCB #1, these two fields could be written as a record as follows:

```
PRINT #1;NAME$,PHONE$
```

NAMES$+PHONE$=1 record. A file consists of one or more records, and a record consists of one or more fields.

The DISKFILE tutorial program demonstrates many of the common functions required in a simple database type program. By using the program and studying the program code, you will learn how datafiles may be handled in ATARI BASIC.

It is important to understand the terminology used here. CREATE means just that. In this case it means create from scratch. Note that the create routine actually begins at line 1000 and that line 1010 contains an OPEN command. The number 8 in that command means write only. If a file is opened using this variable, and a file with the exact same name is found on your diskette, the old file will be deleted automatically.

Using option two, a file is read from disk and displayed on the screen. This does not in any way alter the disk file.

Option three is used to ADD data to an existing disk file only. The term APPEND is often used in this case. In plain English, the term APPEND means, "add to the end of this file."

Option four is used to UPDATE the records of an existing file. This means you will alter, fix, or change a record. This procedure is a bit more complicated than the others since we will not know which record the user may choose to update in advance. The technique used in this demo program is known as Random Access Updating. An index consisting of SECTOR and BYTE locations is created and stored in an array. This gives us the exact spot where each record begins. Since we are using fixed length records of 20 characters each, we can read a specific record into a string, change it in the string, then rewrite the string onto the disk. This becomes a real time saver when many records must be updated in a large disk file.

Option five is used to READ and display a specific file called the DIRECTORY FILE. This DOS generated file contains the table of contents of your diskette. This file is also known as the VTOC or Volume Table Of Contents. For display only, this routine does the same thing as DOS option A.

Although some error trapping has been built in, many possible error conditions are not corrected or fully explained by this program. Error trapping and human engineering account for a great deal of planning and program code. This is not a cop out on my part. The point here is to provide an example of diskfile handling. Accounting for all possible errors could easily double the size of the program, making it more difficult to understand.

That's about it for now. I suggest you use my program as is, then experiment by making minor changes and noteing the results. When you're ready to write your own diskfile handling program, feel free to use these routines.

HOW FILES WAS WRITTEN:

While the rest of this package was designed to provide you with utility programs and related information, the FILES program is not a utility. It was designed to show you how to manipulate data files on disk.

You may be interested in using this information to create your own data file handling program. The following documentation will explain the various BASIC routines used in FILES, so that you may alter and use them in your own database programs.

I wrote FILES using meaningful variable and string names. Look at program lines 100 and 110. DRIVE$ will hold the disk drive number in a format such as "D1:". FILE$ will hold the filename you specify such as "TEST.DAT", and DRIVEFILE$ will hold the combination of DRIVE$ and FILE$ such as "D1:TEST.DAT".

For purposes of this tutorial, we will use very short data records containing 10 characters and use RECORD$ to store the current record. ANSWER$ will be used to store a response from you, the user.

The arrays SECTOR and BYTE will be used to store the location of each data record on the diskette. DIRECTORY$ will store the records we read from the disk directory file.

We setup our menu screen display starting at line 120 and ending at line 210. At the end of line 210, we go to a subroutine which starts at line 7000. This subroutine will be used whenever we need a single character response from the user. Before the INPUT statement, we make sure anything previously stored in ANSWER$ is eliminated and POKE 764,255 to ignore any previously pressed key. In this case, we are waiting for the user to enter the desired option number.

When we return at line 220, we set a trap to line 800 which is an error message routine. Then we store the number 120 in the variable LINE, and the number 6 in the variable HIGHNUMBER. The last thing we do in line 220 is put the numeric value of ANSWER$ into the variable NUMBER.

If the user entered a letter instead of a number, BASIC will give us an error condition because we cannot take the value of a letter. The trap to line 8000 will be activated and the error message from that line will be displayed on the screen. At line 8010, we go to the alarm and time delay subroutine in lines 9000 and 9010, then go back to whatever line number is stored in the variable LINE. In this case that line number is 120.

If the user did enter a number, line 230 checks to make sure the number was not less than 1 or greater than 6, and line 240 sends us to the line where the selected option routine begins.

Let's go through each of the six routines beginning with number one, which begins at line 1000. This routine creates a sample data file consisting of ten records. Notice that we reset the value of LINE to 6100. LINE will always hold the line number that may be used later by a GOTO LINE instruction.

Next we go to the subroutine beginning at line 7100. We will be using this subroutine often, so study it carefully. This routine gets the desired drive number from the user and makes sure it is not less than 1 or greater than 4. Once a valid drive number has been entered, it creates DRIVE$ as we described earlier, and asks for a filename. This information is combined and winds up in the string DRIVEFILE$.

RETURNing to line 1000, we set a trap to line 9100 in case something goes wrong, clear the screen, and attempt to OPEN the file as specified in DRIVEFILE$. We probably didn't have to CLOSE #1 before the OPEN, but it was done just in case that IOCB was left open. In any case, it never hurts.

Notice the operand 8 used in the OPEN command. It is most important that you understand the function of this number. In this case it means we want to OPEN a file to WRITE only. If any other file on the diskette contains the same filename, ATARI's file management system will delete the old file to make room for the new one. It is therefore important to remember that the operand 8 should only be used to create a NEW data file.

Each of the other possibilities for this operand are demonstrated by this program. The number 4 is used if a file is to be OPENed to read only. 9 is used to APPEND data to a file. APPEND means add to the end of an existing file. 12 is used to READ or WRITE and is used for updating an existing file. 6 is used to OPEN the diskette's directory file and read it. By studying the FILES program, you will learn how each of these OPEN operands are used.

The data in our records will consist of ten characters that do nothing but take up space, and show you that we created a ten character record. The program writes ten sample records by using a ? (PRINT) #1 command within a FOR/NEXT loop. We should now have a ten record file to work with, (big deal).

At the end of this routine, we GOTO 6100. This routine closes all IOCBs, sets the print tab width to 5, and displays the message, "PRESS RETURN FOR OPTIONS". This was done to give the user time to read what was displayed on the screen before returning to our main options screen.

There is very little difference between the routine we just used (lines 1000-1100) and the routine beginning at line 2000. Option 2 allows us to read a file like the one we just created. The difference is in the OPEN variable being 4 instead of 8, (read instead of write), and the lack of the FOR/NEXT loop. Since we do not know in advance how many records to read, we just set a trap. When we get an end of file error from BASIC, we know we read the whole file.

The routine beginning at line 3000 is somewhat more complex. Here we will add to our data file. We OPEN using the #9 to APPEND, get record data from the keyboard, add blanks if necessary to insure a record length of 10, and provide an option for the user to add as many records as desired.

Option #4 is by far the most complex. This routine begins at line 4000. Here we OPEN using the #12 to READ/WRITE, and update or change records in an existing data file. First, we have to know the exact location of each record on the diskette. We do this by reading the file and storing the sector and byte location of each record in the appropriately named arrays. The NOTE command in line 4030 stores our current position on the disk. We add 1 to our record counter, then store our position in our arrays. Then we input a record and display all available information on the screen. Once we've read the entire file and we know where each record can be found, we can change any number of records without rereading the file again and again.

To update a record, we just ask the user for the record number to be updated, and POINT to this record as demonstrated in line 4300. The only restriction is that we maintain our fixed record length of 10 positions per record. When we PRINT our updated record on the disk, we will simply be writing the new record where the old one used to be. This process is called random access updating.

In your own programs, you can change the record length to as much as 256 positions, and just modify this demonstation program to store whatever information you desire. If you can stay within 80 characters per record, and you'd like a collection of programs written specifically to provide various types of data storage and retrieval functions along with sorting and printing capability, look into SWIFTWARE's (Swifty Software's) FILEIT 2 package. It can be used to print lists of most anything, address labels, and financial reports. I wrote it with the home user and small businessman in mind.

We still have one more option to describe in addition to #6 which is used to automatically return us to the MENU program. Option 5 reads the disk directory file, and displays it on the screen. This tells us the names of each program and data file on a given diskette along with the number of sectors it occupies, and the number off FREE or unused sectors we have left.

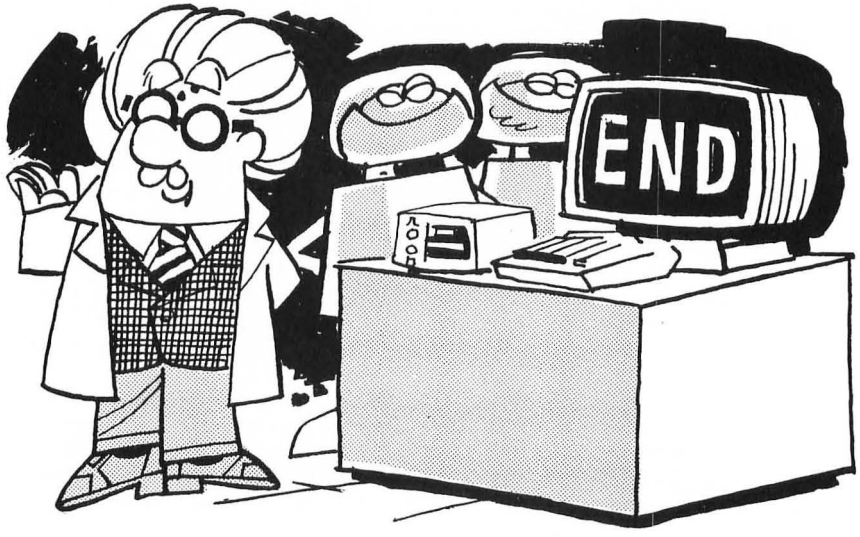
This routine, which begins at line 5000, is very simple. The key is in the OPEN command. The D:*.* specifies disk drive 1, all files. It could also be D1:*.*, but drive one is always assumed if the drive number is not specified. Those asterisks are called wild cards. *.* means all letter/number combinations before and after the period or filename.extention divider. The following are a few

possibilities that could be used to replace the *.*, and what they would specify. See your DOS 2 reference manual for further information.

*. =All filenames with no extentions.
*.SYS =All filenames with the extention .SYS.
M*. =All filenames that begin with M and have no extentions.
M*.* =All filenames that begin with the letter M.

The DISKLIST and MENU programs read this same file to provide their disk jacket label printing and automatic program loading features.

Hopefully, the FILES program listing, this documentation, and the pages that describe how to use this program, combine to provide you with everything you will ever need to know about the handling of data files. As with any form of programming, a little effort and trial and error on your part will be required to put this information into practical use.



This concludes our tutorial on disk files. We hope you find the utility programs to be useful, and learned something about using your disk drive along the way. If you have any comments about this package, feel free to contact the author at the address listed below.

Jerry White
18 Hickory Lane
Levittown, N.Y. 11756

PROGRAM LISTINGS

MENU

```
100 REM MENU (c) 1982 by Jerry White [4/4/82]
110 GOTO 970
120 FOR ME=Q1 TO LEN(B$):IF B$(ME,ME)=" " THEN JW$(ME,ME)="
":GOTO 140
130 JW=ASC(B$(ME,ME))+128:JW$(ME,ME)=CHR$(JW)
140 NEXT ME:RETURN
150 POKE 16,64:POKE 53774,112:POKE 710,14:POKE 708,31:POKE 7
09,172:POKE 711,90:POKE 712,144:RETURN
160 TRAP 170:INPUT #Q1,P$:IF P$(5,Q8)<>"FREE" THEN 260
170 POSITION Q2,Q6:? #Q6;" TOTAL FILES=":POSITION Q2,Q8:? #Q
6;P$(Q1,Q4);"FREE SECTORS":? #Q6;L$:IF BP>Q6 THEN DBP=Q6:TBP
=BP
180 IF BP<=Q6 THEN DBP=BP:TBP=BP
190 BPD$="CONTINUEBASIC    DOS       ":JW$="CONTINUEBASIC    DOS
    ":SEL=Q1:PN=Q3:GOSUB 120
200 POSITION Q6,Q1:? #Q6;"M E N U":POSITION Q7,Q7:? #Q6;"
          "
210 POKE 540,60:POKE Q279,Q3
220 IF PEEK(Q279)=Q6 THEN RETURN
230 IF PEEK(540)>25 THEN POSITION Q4,Q7:? #Q6;"PRESS START":
GOTO 220
240 POSITION Q4,Q7:? #Q6;"              ":IF PEEK(540)=Q3 THEN
210
250 GOTO 220
260 R=R+Q1:POSITION 15,Q6:? #Q6;R:D$(R*17-16,R*17)=P$:IF P$(
11,11)=" " THEN BP=BP+Q1:B$(BP*Q8-7,BP*Q8)=P$(Q7,10)
270 GOTO 160
280 KEY=PEEK(764):POKE 764,255:TEMP=LSEL:LSEL=SEL:KEYSEL=Q1:
IF KEY=31 THEN SEL=Q1:GOTO 650
290 IF KEY=30 THEN SEL=Q2:GOTO 660
300 IF KEY=26 THEN SEL=Q7:GOTO 670
310 IF KEY=24 THEN SEL=Q4:GOTO 380
320 IF KEY=29 THEN SEL=5:GOTO 380
330 IF KEY=27 THEN SEL=Q6:GOTO 380
340 IF KEY=51 THEN SEL=7:GOTO 380
350 IF KEY=53 THEN SEL=Q8:GOTO 380
360 IF KEY=48 THEN SEL=9:GOTO 380
370 LSEL=TEMP:GOTO 590
380 TRAP 40000:KEYSEL=Q1:FP=PN-DBP:CP=SEL-Q7+FP:GOTO 750
390 GRAPHICS 17:GOSUB 150:? #Q6:? #Q6;"    SELECT OPTION":? #
Q6;"     PRESS START":RETURN
400 GRAPHICS 18:GOSUB 150:OP=100:? #Q6;L$:GOSUB 410:GOTO 440

410 POSITION Q4,Q2:? #Q6;"SELECT OPTION":POSITION 5,Q4:? #Q6
;"PRESS START":? #Q6;L$:POSITION Q3,10:? #Q6;L$
420 POSITION Q2,7:? #Q6;"RUN BASIC PROGRAM":POSITION Q2,9:?
#Q6;"DISPLAY DIRECTORY":RETURN
430 POSITION Q2,7:? #Q6;"RUN BASIC PROGRAM":POSITION Q2,9:?
#Q6;"DISPLAY DIRECTORY":RETURN
440 BUT=PEEK(Q279):IF BUT=5 OR BUT=Q7 THEN GOSUB 1050:GOTO 5
00
450 IF BUT=Q6 THEN GOSUB Q20:GOTO 470
460 GOTO 440
470 BUT=PEEK(Q279):IF BUT=Q6 THEN 470
480 GOSUB Q20:IF OP=100 THEN GOSUB 390:GOTO 520
490 GOTO 850
500 IF OP=100 THEN OP=200:GOSUB 430:GOTO 440
510 OP=100:GOSUB 420:GOTO 440
520 IF BP=Q3 THEN 1140
530 IF BP>Q6 THEN DBP=Q6:TBP=BP
540 IF BP<=Q6 THEN DBP=BP:TBP=BP
550 SC=INT(BP/Q6)+Q1:SEL=Q1:PN=Q3
560 S=PEEK(560)+PEEK(561)*256+Q4:SM=PEEK(S)+PEEK(S+Q1)*256:J
```

```
W=USR(ADR(MC$),SM+60,360)
570 FOR DIS=Q1 TO Q7:POSITION Q4,DIS*Q2+Q2:? #Q6;DIS;"= ";BP
D$(DIS*Q8-7,DIS*Q8):NEXT DIS:POSITION Q4,Q4:? #Q6;"1="
580 FOR DIS=Q4 TO DBP+Q7:PN=PN+Q1:POSITION Q4,DIS*Q2+Q2:? #Q
6;DIS;"= ";B$(PN*Q8-7,PN*Q8):NEXT DIS
590 BUT=PEEK(Q279):IF BUT=Q6 THEN GOSUB Q20:GOTO 740
600 IF PEEK(764)<>255 THEN 280
610 IF BUT=Q7 OR BUT=5 THEN 630
620 KEYSEL=Q3:GOTO 590
630 LSEL=SEL:SEL=SEL+Q1:IF SEL=DBP+Q4 THEN SEL=Q1
640 IF SEL>Q7 THEN 710
650 IF SEL=Q1 THEN POSITION Q4,Q4:? #Q6;"1= CONTINUE":POSITI
ON Q4,Q6:? #Q6;"2= BASIC":POSITION Q4,Q8:? #Q6;"3= DOS"
660 IF SEL=Q2 THEN POSITION Q4,Q4:? #Q6;"1= CONTINUE":POSITI
ON Q4,Q6:? #Q6;"2= BASIC":POSITION Q4,Q8:? #Q6;"3= DOS"
670 IF SEL=Q7 THEN POSITION Q4,Q4:? #Q6;"1= CONTINUE":POSITI
ON Q4,Q6:? #Q6;"2= BASIC":POSITION Q4,Q8:? #Q6;"3= DOS"
680 IF KEYSEL THEN 800
690 IF SEL=Q1 THEN FP=PN-DBP:LP=LSEL-Q7+FP:CP=SEL-Q7+FP:GOTO
 720
700 GOSUB 1080:GOTO 590
710 FP=PN-DBP:LP=LSEL-Q7+FP:CP=SEL-Q7+FP:IF SEL=Q4 THEN POSI
TION Q4,Q8:? #Q6;"3= DOS":ISEL=52+128:GOTO 730
720 POSITION Q4,LSEL*Q2+Q2:? #Q6;LSEL;"= ";B$(LP*Q8-7,LP*Q8)
:ISEL=LSEL+49+128:IF SEL=Q1 THEN GOSUB 1080:GOTO 590
730 POSITION Q4,SEL*Q2+Q2:? #Q6;CHR$(ISEL);"= ";JW$(CP*Q8-7,
CP*Q8):GOSUB 1080:GOTO 590
740 BUT=PEEK(Q279):IF BUT=Q6 THEN 740
750 IF SEL<Q4 THEN GOSUB Q20:GOTO 800
760 DOIT$="D1:":TRAP 1160:DOIT$(LEN(DOIT$)+Q1)=B$(CP*Q8-7,CP
*Q8):GRAPHICS 18:GOSUB 150
770 POSITION Q6,Q7:? #Q6;"LOADING":FOR LD=Q1 TO LEN(DOIT$):I
F DOIT$(LD,LD)=" " THEN POP :GOTO 790
780 NEXT LD
790 XX=10-INT(LD/Q2):POSITION XX,5:? #Q6;DOIT$:RUN DOIT$
800 IF SEL=Q2 THEN END
810 IF SEL=Q7 THEN DOS
820 TBP=TBP-Q6:IF TBP<=Q3 THEN GOTO 520
830 IF TBP>=Q6 THEN DBP=Q6:GOTO 560
840 DBP=TBP:GOTO 560
850 TRAP 940:GOSUB 1130:PR=R:SP=Q3:SC=Q3
860 IF PR=Q3 THEN 940
870 SC=SC+Q1:IF SC<10 THEN PR=PR-Q1:SP=SP+Q1:? #Q6:? #Q6;D$(
SP*17-16,SP*17):GOTO 860
880 IF SC=Q3 AND PR=Q3 THEN 940
890 POSITION Q4,22:? #Q6;"PRESS START"
900 IF PEEK(Q279)<>Q6 THEN 900
910 GOSUB Q20
920 IF PEEK(Q279)=Q6 THEN 920
930 GOSUB Q20:SC=Q1:GOSUB 1130:GOTO 860
940 POSITION Q4,22:? #Q6;"PRESS START"
950 IF PEEK(Q279)<>Q6 THEN 950
960 GOSUB Q20:GOTO 1030
970 READ Q1,Q2,Q3,Q4,Q6,Q7,Q8,Q20,Q279
980 DATA 1,2,0,4,6,3,8,1110,53279
990 DIM D$(1088),P$(20),OP$(20),L$(20),BPD$(24),JW$(512),B$(
512),DOIT$(11),MC$(42):L$=""
1000 MC$="hh[illegible]0/[illegible]":GRAPHI
CS 18:GOSUB 150:CLOSE #Q1:OPEN #Q1,Q6,Q3,"D:*.*":? #Q6;L$
1010 POSITION Q6,Q1:? #Q6;"M E N U":POSITION Q7,Q7:? #Q6;"BY
 JERRY WHITE":? #Q6;L$:POKE 580,Q3:POKE 65,Q1
1020 POSITION Q2,Q6:? #Q6;"READING FILE";:GOSUB 160:GOSUB Q2
0
1030 IF PEEK(Q279)=Q6 THEN 1030
1040 GOSUB Q20:GOTO 400
1050 GOSUB Q20
1060 BUT=PEEK(Q279):IF BUT=5 OR BUT=Q7 THEN 1060
1070 RETURN
1080 POKE 540,20
1090 S=PEEK(540):IF S<>Q3 THEN SOUND Q3,SEL-Q1,Q2,5/Q2:GOTO
1090
```

```
1100 SOUND Q3,Q3.Q3,Q3:RETURN
1110 SOUND Q3,102,12,Q8:SOUND Q1,51,12,Q8:POKE Q279,Q3:SOUND
 Q2,Q3,Q2,Q8:SOUND Q7,Q1,Q2,Q8:POKE Q279,Q3
1120 FOR S=Q3 TO Q7:SOUND S,Q3,Q3,Q3:NEXT S:RETURN
1130 GRAPHICS 17:GOSUB 150:? #Q6:? #Q6;"  FILE     EXT SEC":S
C=Q3:RETURN
1140 GRAPHICS Q3:? :? "  THIS PROGRAM REQUIRES THAT BASIC":?
 :? "  PROGRAMS HAVE NO FILE EXTENTIONS."
1150 ? :? "  NO BASIC PROGRAMS WERE FOUND ON":? :? "  THIS D
ISKETTE.":? :END
1160 GRAPHICS 18:GOSUB 150:? #Q6:? #Q6:? #Q6;"     I WAS UNAB
LE":? #Q6:? #Q6;"       TO LOAD ":? #Q6:? #Q6;"     ";DOIT$
1170 FOR JW=100 TO 255:SOUND Q3,JW,10,Q8:SOUND Q1,JW-Q2,10,Q
8:NEXT JW:SOUND Q3,Q3,Q3,Q3:SOUND Q1,Q3,Q3,Q3:TRAP 40000:GOT
O 400
```

FORMAT1

```
0 REM FORMAT1 (c) 1982 BY JERRY WHITE (DISK FORMATTING UTILI
TY AND TUTORIAL)
1 REM
9 REM SET TEXT MODE, BLACK & WHITE, MARGINS, PRINTTAB WIDTH,
 NO CURSOR.
10 GRAPHICS 0:SETCOLOR 2,0,0:POKE 82,2:POKE 83,39:POKE 201,1
0:POKE 752,1:?
20 ? ,"FORMAT DISK ON DR.1":? :? "          PRESS START WHEN R
EADY.":GOTO 80
30 REM STORE CURRENT SUCCESSFUL FORMATS+1 THEN SAY WHAT WE A
RE UP TO.
40 F=F+1:? CHR$(125):? :? :? :? ,"FORMATTING DISK #";F
50 TRAP 200:XIO 254,#1,0,0,"D1:":TRAP 40000:REM SET DRIVE#1 &
 START FORMATTING
55 POKE 66,1:? CHR$(253):POKE 66,0:REM RING THE BELL BUT NOT
 TOO LOUD
60 ? ,"DISK FORMATTED":? :POKE 77,0:REM DISPLAY RESULT & KIL
L ATTRACT MODE
70 ? " PRESS START TO FORMAT ANOTHER DISK":? " PRESS OPTION
TO END THIS PROGRAM"
80 IF PEEK(53279)=6 THEN 40:REM INSTANT REPLAY.
90 IF PEEK(53279)=3 THEN 300:REM THAT'S ALL FOLKS.
100 GOTO 80:REM WHATCHA WANT? PRESS A BUTTON!
180 REM A TRAP TO LINE 200 WAS SET IN LINE 50 FOR UNFORMATTA
BLE DISKS
190 REM ABORT BY TAPPING THE BREAK KEY (IN MOST CASES)
195 REM YOU MIGHT HAVE TO REMOVE THE DISK, THEN TURN THE DRI
VE OFF & ON
200 ? CHR$(253):? :? "I WAS UNABLE TO FORMAT THAT DISKETTE":
?
210 BAD=BAD+1:F=F-1:GOTO 70:REM ONE LESS SUCCESSFUL FORMAT
300 ? CHR$(125):? :? F;" DISKS FORMATTED":? :? "UNABLE TO FO
RMAT ";BAD;" DISK(S)"
310 REM SAY WHAT WE ACCOMPLISHED THEN WAIT FOR THE USER TO P
RESS A KEY
320 ? :? "          END OF PROGRAM FORMAT1":? :? "          PRES
S ANY KEY FOR MENU":POKE 764,255
330 IF PEEK(764)=255 AND PEEK(53279)=7 THEN 330
340 POKE 764,255:RUN "D:MENU"
```

DISKLIST

```
0 REM DISKLIST 4/2/82 (c) 1982 by Jerry White
110 GRAPHICS 0:POKE 752,1:SETCOLOR 2,5,0:DIM A$(40),DN$(40),
F$(10),B$(40):B$(1)=" ":B$(40)=" ":B$(2)=B$(1)
120 ? "K↓↓            ┌──────────────────────┐":? "          D I S
K L I S T ▌":? "           └──────────────────────┘"
122 FOR ME=15 TO 0 STEP -0.2:SOUND 0,51,12,ME:NEXT ME
130 ? "↓↓↓This program will print a diskette"
140 ? :? "directory that may be attached to a"
150 ? :? "disk jacket.  Insert your diskette,":FOR ME=15 TO
0 STEP -0.2:SOUND 0,102,12,ME:NEXT ME
160 ? :POKE 752,0:? "then type the drive number";:TRAP 120:I
NPUT DN:TRAP 40000
180 ? "K↓↓↓↓TYPE A HEADING IDENTIFICATION LINE↓↓":INPUT DN$
200 F$="D":F$(LEN(F$)+1)=STR$(DN):F$(LEN(F$)+1)=":*.*":CLOSE
 #1:TRAP 600:OPEN #1,6,0,F$
202 TRAP 1000:CLOSE #2:OPEN #2,8,0,"P:":TRAP 40000:? "K↓↓↓↓
        W O R K I N G   ":POKE 752,1:? " "
204 LDN=LEN(DN$):IF LDN=0 OR LDN>33 THEN 208
206 TRAP 208:JW=INT((40-LDN)/2):PRINT #2;B$(1,JW+1);DN$:TRAP
 40000:GOTO 210
208 PRINT #2;"     ";DN$
210 PRINT #2:? #2;"    FILE    EXT SEC    FILE    EXT SEC":?
 #2
220 TRAP 250:C=C+1:SETCOLOR 4,C,8:INPUT #1,A$:IF LEN(A$)<5 O
R A$(5,8)="FREE" THEN 250
230 ? #2;"  ";A$;:C=C+1:SETCOLOR 4,C,8:INPUT #1,A$:IF LEN(A$
)<5 OR A$(5,8)="FREE" THEN 250
240 ? #2;"  ";A$:GOTO 220
250 ? #2:? #2;"           ";A$(1,3);" FREE BLOCKS AVAILABLE.":F
OR I=1 TO 6:PRINT #2:NEXT I:CLOSE #2
260 ? "K↓↓↓↓":SETCOLOR 2,15,0:? "      END OF JOB DISKLIST
 "
280 ? :? "         PRESS OPTION FOR MENU"
290 ? :? "         PRESS START  TO RERUN"
300 IF PEEK(53279)=6 THEN RUN
310 IF PEEK(53279)=3 THEN ? "K":? :? ,"LOADING MENU...":TRAP
 500:RUN "D:MENU"
320 GOTO 300
500 ? "K↓↓↓↓   I COULDN'T LOAD THE PROGRAM ":FOR ME=1 TO 20
0:NEXT ME
510 ? :? "   PLEASE INSERT A DISKETTE INTO":? :? "    DRIVE O
NE WHICH CONTAINS THE":? :? "    MENU PROGRAM."
520 ? "↓↓↓        PRESS START WHEN READY"
530 IF PEEK(53279)=6 THEN TRAP 500:RUN "D:MENU"
540 GOTO 530
600 TRAP 40000:? "K↓↓↓↓    I COULDN'T READ THAT DISKETTE ":FO
R ME=1 TO 500:NEXT ME:RUN
1000 ? "K↓↓↓↓        YOUR PRINTER ISN'T READY":? :? "       PRES
S START WHEN READY..."
1010 IF PEEK(53279)<>6 THEN 1010
1020 GOTO 202
```

AUTOBOOT

```
100 REM AUTOBOOT (CREATES AUTORUN.SYS FILE) (c) 1982 Jerry W
hite
110 DIM A$(128),FN$(14):FN$="D1:AUTORUN.SYS":GOTO 280
120 FOR ME=1 TO 88:READ IT:PUT #1,IT:NEXT ME:RETURN
130 FOR ME=1 TO 20:READ IT:PUT #1,IT:NEXT ME:RETURN
140 FOR I=1 TO 123:READ D:IF I=64 THEN PUT #1,LEN(A$)-1:GOTO
 160
```

```
150 PUT #1,D
160 NEXT I
170 FOR I=LEN(A$) TO 1 STEP -1:PUT #1,ASC(A$(I,I)):NEXT I
180 PUT #1,255:PUT #1,255:PUT #1,226:PUT #1,2:PUT #1,227:PUT
 #1,2:PUT #1,0:PUT #1,6
190 ? :? " CLOSING  ";FN$:CLOSE #1:POKE 82,2:POKE 83,39:? :P
OKE 752,0:END
200 DATA 255,255,0,6,5,6,169,148,141,197,2,96,255,255,226,2,
227,2,0,6
210 DATA 255,255,0,56,75,56,169,80,141,0,3,169,1,141,1,3,169
,63,141,2,3,169,64,141,3,3,169,5,141,6
220 DATA 3,141,5,3,169,0,141,4,3,141,9,3,141,10,3,141,11,3,1
69,12,141,8,3,32,89,228,16,1,96,162
230 DATA 11,189,0,5,157,0,3,202,16,247,32,89,228,48,6,32,6,5
,108,12,0,96,226,2,227,2,0,56
240 DATA 162,0,189,26,3,201,69,240,5,232,232,232,208,244,232
,142,105,6,189,26,3,133,205,169,107,157,26,3,232,189
250 DATA 26,3,133,206,169,6,157,26,3,160,0,162,16,177,205,15
3,107,6,200,202,208,247,169,67,141,111,6,169,6,141
260 DATA 112,6,169,10,141,106,6,96,172,106,6,48,9,185,123,6,
206,106,6,160,1,96,138,72,174,105,6,165,205,157
270 DATA 26,3,232,165,206,157,26,3,104,170,169,155,160,1,96,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,76,0,0,0
280 GRAPHICS 0:POKE 710,144:POKE 82,1:POKE 83,38:POKE 201,10
:? :? " AUTORUN.SYS CREATE PROGRAM FOR BASIC"
290 ? ,"  DOS VERSION 2":? :? "    UPDATED 3/25/82 BY JERRY
WHITE"
300 ? :? " BOOT INTERFACE (Y or N)?";:GOSUB 360
310 ? :? " ENTER COMMANDS":? :INPUT A$:POKE 752,1
320 ? :? " OPENING  ";FN$:CLOSE #1:OPEN #1,8,0,FN$:? :? " CR
EATING ";FN$
330 IF BOOT850 THEN RESTORE 200:GOSUB 130:RESTORE 210:GOSUB
120:GOTO 350
340 RESTORE 200:GOSUB 130
350 PUT #1,255:PUT #1,255:PUT #1,0:PUT #1,6:L=123+LEN(A$)-1:
PUT #1,L:PUT #1,6:RESTORE 240:GOTO 140
360 POKE 764,255
370 IF PEEK(764)=255 THEN 370
380 IF PEEK(764)=43 THEN ? "Y":BOOT850=1:POKE 764,255:RETURN

390 IF PEEK(764)=35 THEN ? "N":BOOT850=0:POKE 764,255:RETURN

400 ? :? :? "           PLEASE TYPE Y OR N?";:GOTO 360
500 ? "■↓↓ I WAS UNABLE TO OPEN AUTORUN.SYS":GOTO 900
600 ? "■↓↓ ERROR WHILE WRITING AUTORUN.SYS"
900 ? :? ,"ERROR NUMBER ";PEEK(195):POKE 201,8:? :? ,"PRESS
OPTION TO RERUN":? ,"PRESS START FOR DOS";
910 IF PEEK(53279)=3 THEN RUN
920 IF PEEK(53279)=6 THEN CLOSE #1:DOS
930 GOTO 910
```

INSPECT

```
100 REM INSPECT (c) 1982 Jerry White [3/12/82]
110 GOTO 940
120 REM
130 REM **  DISPLAY A SECTOR IN CHARACTER FORMAT **
140 REM
150 FOR X=A TO B:IF SEC$(X,X)=CHR$(155) THEN ? " ";:GOTO 170
160 ? SEC$(X,X);
170 NEXT X:RETURN
180 REM
190 REM ** DISPLAY A SECTOR IN HEX FORMAT **
200 REM
```

```
210 FOR X=A TO B:BYTE=ASC(SEC$(X,X)):HX=INT(BYTE/16)+1:HEX=B
YTE-16*(HX-1)+1:HEX$(1,1)=HEXSTR$(HX,HX)
220 HEX$(4,4)=HEXSTR$(HEX,HEX):? HEX$;:NEXT X:RETURN
230 REM
240 REM ** EXAMINE DIRECTORY FILE **
250 REM
260 POKE 764,255:POKE 82,2:? "}":? "S FN FILE     EXT      SSEC
      NSEC":?
270 POKE 779,1:FOR LOOP=1 TO 8:POKE 778,104+LOOP:IO=USR(ADR(
CIO$))
280 FOR ENTRY=1 TO 8:STATE$=SEC$(ENTRY*16-15,ENTRY*16-14):IF
 STATE$="♥♥" THEN 340
290 IF STATE$(1,1)="♥" THEN ? "D ";:GOTO 320
300 LOCK=ASC(STATE$(1,1)):LOCK=LOCK-INT(LOCK/64)*64:IF LOCK>
=32 THEN ? "L ";:GOTO 320
310 ? "  ";
320 ? 8*(LOOP-1)+ENTRY;"▶";SEC$(ENTRY*16-10,ENTRY*16);"▶";
330 ? ASC(SEC$(ENTRY*16-12))+256*ASC(SEC$(ENTRY*16-11));"▶";
ASC(SEC$(ENTRY*16-14))+256*ASC(SEC$(ENTRY*16-13))
340 NEXT ENTRY:NEXT LOOP:GOTO 750
350 REM
360 REM ** EXAMINE SECTOR **
370 REM
380 POKE 764,255:GOSUB 820:POKE 82,1
390 ? "}SECTOR NUMBER ";NUM:? :IO=USR(ADR(CIO$)):IF C=1 THEN
 POKE 766,1
400 ? "BYTE: 000000000111111111122222222223":? "      0123
4567890123456789012345678901"
410 ? "          ";:A=1:B=32:IF C=1 THEN GOSUB 150:GOTO 430
420 GOSUB 210
430 ? :? :? :? "BYTE: 3333333444444444455555555556666":? "
      234567890123456789012345678901234"
440 ? "          ";:A=33:B=64:IF C=1 THEN GOSUB 150:GOTO 460
450 GOSUB 210
460 ? :? :? :? "BYTE: 6666667777777777888888888899999999":? "
      4567890123456789012345678901234 5"
470 ? "          ";:A=65:B=96:IF C=1 THEN GOSUB 150:GOTO 490
480 GOSUB 210
490 ? :? :? :? "BYTE:      11111111111111111111111111111111"
500 ? "      9999000000000011111111112222222222"
510 ? "      67890123456789012345678901234567"
520 ? "          ";:A=97:B=128:IF C=1 THEN GOSUB 150:GOTO 540
530 GOSUB 210
540 POKE 766,0:? :GOTO 750
550 REM
560 REM ** EXAMINE FILE **
570 REM
580 POKE 764,255:TRAP 580:? "}":? :? "        ENTER FILENAME";:
INPUT USER$:FILE$=DRIVE$:FILE$(LEN(FILE$)+1)=USER$
590 TRAP 690:CLOSE #1:OPEN #1,4,0,FILE$:COUNT=0:SCOUNT=0:POK
E 201,10:POKE 82,3:POKE 83,39:FLAG=1
600 POKE 752,1:POKE 766,0:? "}":POKE 766,1:? "BYTE   DEC VALU
E HEX VALUE   CHAR"
610 GET #1,BYTE:COUNT=COUNT+1:SCOUNT=SCOUNT+1:HX=INT(BYTE/16
)+1:HEX=BYTE-16*(HX-1)+1:HEX$(1,1)=HEXSTR$(HX,HX)
620 HEX$(4,4)=HEXSTR$(HEX,HEX):? " ";COUNT,BYTE,HEX$(1,1);HE
X$(4,4);:IF BYTE=155 THEN BYTE=32
630 ? "      ";CHR$(BYTE)
640 IF SCOUNT=20 THEN GOSUB 1230:SCOUNT=0:GOTO 600
650 GOTO 610
660 REM
670 REM ** TRAP ERRORS **
680 REM
690 POKE 66,0:ERROR=PEEK(195):IF ERROR=136 THEN ? :? ,"END O
F FILE":? :? :GOSUB 750:RUN
700 IF ERROR=170 THEN ? :? ,"FILE NOT FOUND":? :? :GOTO 750
710 ? :? ,"ERROR ";ERROR;" AT LINE ";PEEK(186)+PEEK(187)*256
:? :? :GOTO 750
720 REM
```

```
730 REM ** TEMPORARY HALT **
740 REM
750 POSITION 7,23:? "PRESS ANY KEY FOR OPTIONS";:POKE 764,25
5
760 IF PEEK(764)=255 THEN 760
770 IF FLAG THEN RETURN
780 POKE 764,255:GOTO 1040
790 REM
800 REM ** INPUT SECTOR NUMBER & FORMAT **
810 REM
820 TRAP 820:? :? "     TYPE SECTOR NUMBER RETURN";:INPUT NUM
:TRAP 40000
830 NUM=INT(NUM):IF NUM<1 OR NUM>720 THEN ? :? ,"ENTER 1 THR
U 720":GOTO 820
840 ? :? "TYPE 1 FOR CHARACTER OR 2 FOR HEXDUMP";:POKE 764,2
55
850 LASTKEY=PEEK(764):IF LASTKEY=255 THEN 850
860 IF LASTKEY=31 THEN C=1:GOTO 890
870 IF LASTKEY=30 THEN C=2:GOTO 890
880 ? :GOTO 840
890 POKE 778,NUM-INT(NUM/256)*256
900 POKE 779,INT(NUM/256):RETURN
910 REM
920 REM ** PROGRAM SETUP **
930 REM
940 DIM CIO$(5),SEC$(128),STATE$(2),HEX$(5),HEXSTR$(16):CIO$
="h 5"
950 HEX$=" ↓← ↑":HEXSTR$="0123456789ABCDEF"
960 DIM DRIVE$(3),USER$(12),FILE$(15):DRIVE$="D1:"
970 POKE 772,ADR(SEC$)-INT(ADR(SEC$)/256)*256:REM LOW BYTE
980 POKE 773,INT(ADR(SEC$)/256):REM HIGH BYTE
990 POKE 769,1:REM DRIVE 1
1000 POKE 770,82:REM READ SECTOR (87=WRITE SECTOR)
1010 REM
1020 REM ** MATCH-MAKER DO **
1030 REM
1040 GRAPHICS 0:POKE 710,144:POKE 201,9:POKE 752,1:POKE 82,2
:POKE 83,39:FLAG=0:POKE 766,0
1050 ? :? ,"┌──────────────────────┐"
1060 ? ,"| DISK INSPECTOR |"
1070 ? ,"└──────────────────────┘"
1080 ? :? :? ,"TYPE OPTION NUMBER"
1090 ? :? ,"1= EXAMINE DIRECTORY"
1100 ? :? ,"2= EXAMINE SECTOR"
1110 ? :? ,"3= EXAMINE FILE"
1120 ? :? ,"4= RUN MENU"
1130 POKE 764,255:SEC$(1)=" ":SEC$(128)=" ":SEC$(2)=SEC$(1)
1140 LASTKEY=PEEK(764):IF LASTKEY=255 THEN 1140
1150 IF LASTKEY=31 THEN 240
1160 IF LASTKEY=30 THEN 360
1170 IF LASTKEY=26 THEN POKE 752,0:GOTO 560
1180 IF LASTKEY=24 THEN POKE 201,10:GRAPHICS 0:POKE 752,1:PO
KE 764,255:? :? :? ,"LOADING MENU":RUN "D:MENU"
1190 GOTO 1130
1200 REM
1210 REM ** START OR OPTIONS **
1220 REM
1230 POSITION 5,23:? "OPTIONS=OPTIONS  START=CONTINUE";
1240 IF PEEK(53279)=3 THEN RUN
1250 IF PEEK(53279)=6 THEN RETURN
1260 GOTO 1240
```

SPEEDCHECK

```
0 REM RPMTEST 4/29/82
100 GRAPHICS 0:POKE 559,0:FOR I=0 TO 72:READ A:POKE 1536+I,A
:NEXT I:POKE 710,0:POKE 82,1:POKE 201,10
120 ? "K":? ,"┌──────────────────────────────┐":? ,"│ R P M T E S T │":? ,"
└──────────────────────────────┘":POKE 559,34:POKE 752,0
150 ? "↓↓↓▶TEST WHICH DRIVE NUMBER";:TRAP 150:INPUT DRIVE:TR
AP 40000
160 DRIVE=INT(DRIVE):IF DRIVE<1 OR DRIVE>4 THEN ? :? "   ENT
ER DRIVE NUMBER 1, 2, 3, OR 4":GOTO 150
180 POKE 752,1:? :? "     TESTING 100 DISK REVOLUTIONS":? :?
 "  THIS SHOULD TAKE APPROX. 22 SECONDS";
200 POKE 1610,DRIVE:X=USR(1536):A=PEEK(1611):B=PEEK(1612):MI
N=(256*B+A)/3600:RPM=INT(100/MIN+0.5)
220 ? :? :? "▶▶RPM=";RPM:IF RPM>284 AND RPM<291 THEN ? :? "▶
   DRIVE SPEED IS O.K.":POKE 710,178:GOTO 250
230 POKE 710,66:IF RPM<285 THEN ? :? "▶DRIVE SPEED IS TOO SL
OW":GOTO 250
240 ? :? "▶DRIVE SPEED IS TOO FAST"
250 ? :? "PRESS OPTION FOR MENU OR START TO RERUN";
300 IF PEEK(53279)=6 THEN 120
350 IF PEEK(53279)=3 THEN POKE 82,2:? "K":? ," LOADING MENU"
:RUN "D:MENU"
400 GOTO 300
500 REM DATA FOR MACHINE LANGUAGE ROUTINE
1000 DATA 104,169,1,141,10,3,169,0
1010 DATA 141,11,3,141,4,3,169,5
1020 DATA 141,5,3,173,74,6,141,1
1030 DATA 3,169,82,141,2,3,169,5
1040 DATA 141,73,6,32,83,228,206,73
1050 DATA 6,208,248,169,100,141,73,6
1060 DATA 169,0,133,19,133,20,32,83
1070 DATA 228,206,73,6,208,248,165,20
1080 DATA 164,19,141,75,6,140,76,6,96
2000 REM
2010 REM THIS IS A MODIFIED VERSION OF THE RPMTEST PROGRAM O
RIGINALLY PUBLISHED
2020 REM IN THE MAY 1982 ISSUE OF COMPUTE! MAGAZINE.
2030 REM WE WISH TO THANK BOB CHRISTIANSEN FOR HIS PERMISSIO
N TO INCLUDE THIS
2040 REM PROGRAM AS PART OF THIS TUTORIAL/UTILITY PACKAGE.
```

DISKFILES

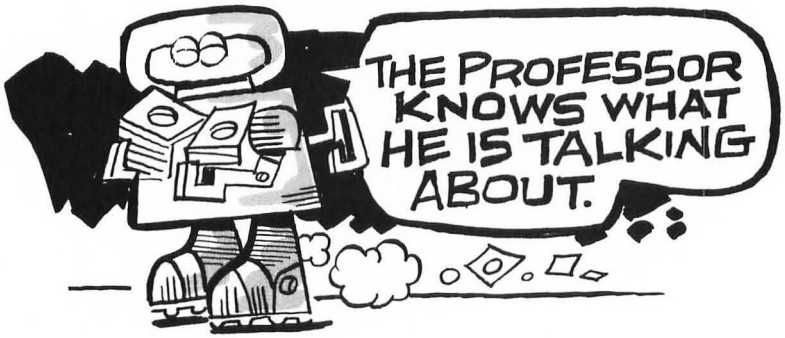
```
0 REM FILES (c) 1981 by Jerry White
1 REM ATARI DISKFILE TUTORIAL DEMO
2 REM
100 DIM DRIVE$(3),FILE$(12),DRIVEFILE$(15),RECORD$(10),ANSWE
R$(1)
110 DIM SECTOR(20),BYTE(20),DIRECTORY$(20):REM DIMENSION STR
INGS AND ARRAYS
111 REM
120 GRAPHICS 0:POKE 82,2:POKE 83,39:REM CLEAR SCREEN AND SET
 MARGINS
130 POKE 201,5:SETCOLOR 2,1,0:REM SET PRINT TAB WIDTH TO 5 S
PACES AND COLOR
140 ? :? "TYPE OPTION NUMBER THEN PRESS RETURN"
150 ? :? ,"(1) CREATE A DISK FILE":REM GOTO 1000
160 ? :? ,"(2) READ A DISK FILE":REM GOTO 2000
170 ? :? ,"(3) ADD TO A DISK FILE":REM GOTO 3000
180 ? :? ,"(4) UPDATE A DISK FILE":REM GOTO 4000
190 ? :? ,"(5) DISPLAY DISK DIRECTORY":REM GOTO 5000
200 ? :? ,"(6) RUN MENU":REM GOTO 9140
210 ? :? ,"YOUR CHOICE";:GOSUB 7000
220 TRAP 8000:LINE=120:HIGHNUMBER=6:NUMBER=VAL(ANSWER$)
230 IF NUMBER<1 OR NUMBER>6 THEN GOTO 8000
240 ON NUMBER GOTO 1000,2000,3000,4000,5000,9140
```

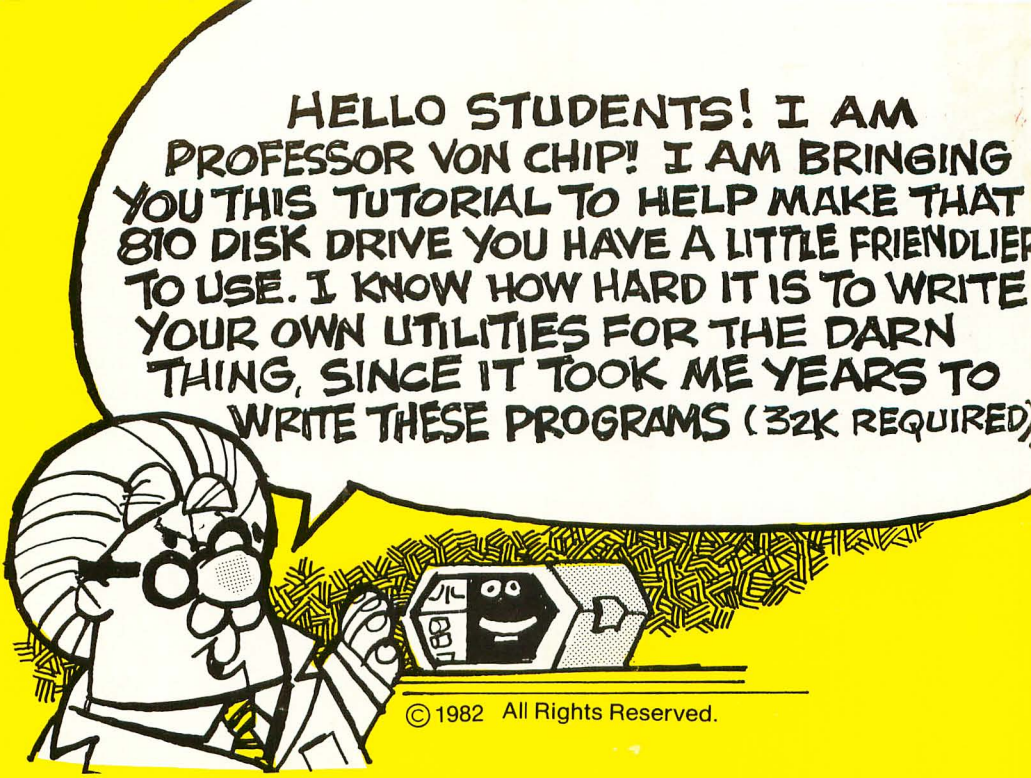
```
250 REM
1000 LINE=6100:GOSUB 7100:TRAP 9100:GRAPHICS 0:SETCOLOR 2,4,
0
1010 CLOSE #1:OPEN #1,8,0,DRIVEFILE$
1020 ? :? "CREATING ";DRIVEFILE$:? :RECORD$="1234567890"
1030 FOR DEMO=1 TO 10
1040 ? #1;RECORD$
1050 ? "WRITING RECORD NUMBER ";DEMO
1060 NEXT DEMO
1070 ? :? "10 RECORD DEMO FILE CREATED"
1080 ? :? "CLOSING ";DRIVEFILE$
1090 CLOSE #1
1100 GOTO 6100
1110 REM
2000 LINE=6100:GOSUB 7100:TRAP 9100:GRAPHICS 0:SETCOLOR 2,5,
0:? :?
2010 CLOSE #2:OPEN #2,4,0,DRIVEFILE$:RECNUM=0:LINE=6100
2020 INPUT #2,RECORD$
2030 RECNUM=RECNUM+1
2040 ? "RECORD NUMBER ";RECNUM;
2050 ? ,RECORD$
2060 GOTO 2020
2070 REM
3000 LINE=3000:GOSUB 7100:TRAP 9100:GRAPHICS 0:SETCOLOR 2,7,
0
3010 CLOSE #3:OPEN #3,9,0,DRIVEFILE$
3020 GRAPHICS 0:? :? ,"ADD RECORD(S) ROUTINE:"
3030 ? :? ,"ENTER 10 CHARACER RECORD"
3040 ? :? ,"OR JUST PRESS RETURN TO EXIT":? :GOSUB 6000
3050 RECLEN=LEN(RECORD$):IF RECLEN=0 THEN 3200
3060 IF RECLEN=10 THEN 3090
3070 FOR BLANK=RECLEN+1 TO 10:RECORD$(LEN(RECORD$)+1)=" ":NE
XT BLANK
3090 PRINT #3;RECORD$
3100 ? :? "PRESS START TO ENTER ANOTHER RECORD"
3110 ? :? "PRESS OPTION FOR OTHER OPTIONS...";
3120 IF PEEK(53279)=6 THEN 3020
3130 IF PEEK(53279)=3 THEN 3200
3140 GOTO 3120
3200 ? :? :? ,"ADDING RECORD(S) TO DISK":CLOSE #3:GOTO 120
3210 REM
4000 LINE=4100:GOSUB 7100:TRAP 9100:GRAPHICS 0:SETCOLOR 2,10
,0
4010 CLOSE #4:OPEN #4,12,0,DRIVEFILE$:LINE=4100
4020 ? :? ,,"CREATING INDEX":RECNUM=0
4030 NOTE #4,SECTOR,BYTE
4040 RECNUM=RECNUM+1
4050 SECTOR(RECNUM)=SECTOR:BYTE(RECNUM)=BYTE
4060 INPUT #4,RECORD$:? ," RECORD ";RECNUM,RECORD$
4070 ? ,"SECTOR=";SECTOR,"BYTE=";BYTE
4080 ? :GOTO 4030
4100 RECNUM=RECNUM-1
4110 ? :? "PRESS START TO UPDATE A RECORD"
4120 ? :? "PRESS OPTION FOR OTHER OPTIONS";
4130 IF PEEK(53279)=6 THEN 4200
4140 IF PEEK(53279)=3 THEN CLOSE #4:GOTO 120
4150 GOTO 4130
4200 GRAPHICS 0:REM RANDOM ACCESS RECORD UPDATE ROUTINE
4210 ? :? ,"DISKFILE CONTAINS ";RECNUM;" RECORDS"
4220 ? :? "ENTER RECORD NUMBER TO BE UPDATED";
4230 TRAP 4220:INPUT UPDATE:TRAP 40000
4240 UPDATE=INT(UPDATE):IF UPDATE<1 OR UPDATE>RECNUM THEN 42
30
4250 POINT #4,SECTOR(UPDATE),BYTE(UPDATE)
4260 INPUT #4,RECORD$:? :? RECORD$
4270 ? :? "ENTER NEW RECORD #";UPDATE;:INPUT RECORD$
4280 RECLEN=LEN(RECORD$):IF RECLEN=10 THEN 4300
4290 FOR BLANK=RECLEN+1 TO 10:RECORD$(LEN(RECORD$)+1)=" ":NE
XT BLANK
4300 POINT #4,SECTOR(UPDATE),BYTE(UPDATE)
4310 PRINT #4;RECORD$:? :? ,"RECORD HAS BEEN UPDATED"
4320 GOTO 4110
```

```
4330 REM
5000 GRAPHICS 0:POKE 752,1:SETCOLOR 2,12,0:POKE 201,8:? :? ,
"  DISK DIRECTORY":? :TRAP 9100
5010 CLOSE #5:OPEN #5,6,0,"D:*.*":REM OPEN DISK DIRECTORY FO
R ALL ENTRIES
5020 LINE=6100
5030 INPUT #5,DIRECTORY$
5040 ? ,DIRECTORY$
5050 GOTO 5030
5060 REM
6000 RECORD$="":POKE 764,255:REM RECORD STRING AND LAST KEY
PRESSED=NULL
6010 INPUT RECORD$:RETURN
6020 REM
6100 FOR FILE=1 TO 5:CLOSE #FILE:NEXT FILE:REM CLOSE ALL FIL
ES
6110 POKE 201,5:? :? ,"PRESS RETURN FOR OPTIONS";
6120 GOSUB 7000:GOTO 120:REM PAUSE TO READ SCREEN THEN GO TO
 OPTIONS
6130 REM
7000 ANSWER$="":POKE 764,255:INPUT ANSWER$:RETURN :REM 1 CHA
RACTER INPUT
7010 REM
7100 GRAPHICS 0:SETCOLOR 2,0,0:REM DRIVE NUMBER AND FILENAME
 INPUT ROUTINE
7110 ? :? "TYPE DISK DRIVE NUMBER (1-4)";:HIGHNUMBER=4:GOSUB
 7000
7120 LINE=7110:TRAP 8000:NUMBER=VAL(ANSWER$):TRAP 9100
7130 IF NUMBER<1 OR NUMBER>4 THEN 8000
7140 DRIVE$="D":DRIVE$(LEN(DRIVE$)+1)=ANSWER$
7150 DRIVE$(LEN(DRIVE$)+1)=":"
7200 ? :? "TYPE FILE NAME";:INPUT FILE$:IF LEN(FILE$)=0 THEN
 7200
7210 DRIVEFILE$=DRIVE$
7220 DRIVEFILE$(LEN(DRIVEFILE$)+1)=FILE$:RETURN
7230 REM
8000 ? :? "PLEASE TYPE A NUMBER FROM 1 THRU ";HIGHNUMBER:REM
 ERROR ROUTINE
8010 GOSUB 9000:GOTO LINE:REM GO BACK TO LINE NUMBER (LINE)
9000 ? CHR$(253):REM RING ERROR BELL
9010 FOR COUNT=1 TO 300:NEXT COUNT:RETURN
9020 REM
9100 POKE 752,0:IF PEEK(195)=136 THEN GOTO LINE:REM ERROR WA
S END OF FILE
9110 REM DISPLAY ERROR NUMBER AND LINE AT WHICH ERROR OCCURR
ED THEN END
9120 ? :? " ERROR ";PEEK(195);" AT LINE ";PEEK(186)+PEEK(187
)*256
9130 LIST PEEK(186)+PEEK(187)*256:GOSUB 9000
9140 TRAP 40000:GRAPHICS 0:POKE 764,255:POKE 752,1:POKE 201,
10
9150 ? :? :? ,"LOADING MENU":RUN "D:MENU"
```

Don't forget to ask for my other lessons teaching you about all of the great GRAPHICS and SOUND tricks that the ATARI computers can do-the TRICKY TUTORIALS™ for 16K:

#1 — DISPLAY LISTS (many Graphics Modes at the same time!)
#2 — SCROLLING (move your Graphics and text around smoothly)
#3 — PAGE FLIPPING (a professional looking way to redraw many screens of text or graphics)
#4 — BASICS OF ANIMATION (a beginner's lesson in moving shapes around the screen)
#5 — PLAYER MISSILE GRAPHICS (Learn to write a PACMAN™ type game of your own!)
#6 — SOUND AND MUSIC (from single notes & chords to songs & special effects, I explain all)
#7 — DISK UTILITIES (Utility programs to help you use your disk drives (32K)
#8 & 9 Coming soon to a blackboard near you!





Here are the utilities on this disk:

1) A **MENU** program to tell you what is on each disk, and allow you to run the program of your choice at the touch of a button.
2) A **FORMAT** utility that allows you to format your disks with the touch of a single button (sound familiar? I like buttons).
3) Would you like a listing of the contents of each disk? If you have a printer, or can borrow one, that is exactly what you will get with **DISKLIST.** Tape the listing to each jacket cover and you are all set.
4) With **AUTOBOOT,** your disks will run themselves when you turn on the computer. You simply give the computer a list of instructions to follow when it first runs.
5) **INSPECT** will make you as smart as a Professor! You can look at any place on your disks and see what is written there.
6) Finally I give you **DISKFILE,** a small tutorial on writing and reading information to a disk. Perfect for those of you who want to really understand the mysteries of DISKOLOGY (one of my favorite subjects)!
7) A bonus! My Robot, Prototype, has improved a recent program published in COMPUTE! (of course I asked first). **SPEEDCHECK** will help you be sure your drive is running at the right speed.